Dataset yang digunakan terdiri dari 1.000 sampel dengan dua fitur numerik sebagai variabel input dan satu label kelas biner sebagai variabel target. Label kelas diberikan berdasarkan jumlah kedua fitur, di mana kelas 1 diberikan jika jumlah fitur lebih besar dari 1, dan kelas 0 diberikan jika sebaliknya.

Desain Eksperimen

Eksperimen ini bertujuan untuk mengevaluasi pengaruh kombinasi berbagai parameter berikut terhadap akurasi model:

1. **Arsitektur Hidden Layer**:

- Terdapat tiga konfigurasi hidden layer yang diuji: satu layer dengan 4 neuron, dua layer dengan 8 neuron di setiap layer, dan tiga layer dengan 16 neuron di setiap layer.
- Variasi ini bertujuan untuk mengamati pengaruh kompleksitas arsitektur terhadap kemampuan model dalam menangkap pola pada data.

2. **Fungsi Aktivasi**:

- Fungsi aktivasi yang diuji adalah *linear*, *sigmoid*, *ReLU*, *softmax*, dan *tanh*.
- Fungsi aktivasi memengaruhi bagaimana informasi diteruskan dari satu layer ke layer berikutnya dan berperan penting dalam menentukan kemampuan model untuk menangkap non-linearitas data.

3. **Jumlah Epoch**:

- Model dilatih selama berbagai jumlah epoch, mulai dari 1 hingga 250 epoch.
- Eksperimen ini dirancang untuk memahami bagaimana peningkatan jumlah iterasi pelatihan memengaruhi konvergensi model dan menghindari *underfitting* atau *overfitting*.

4. **Learning Rate**:

- Learning rate diuji dengan nilai 10, 1, 0.1, 0.01, 0.001, dan 0.0001.
- Learning rate menentukan seberapa besar langkah pembaruan parameter model selama proses pelatihan. Nilai yang terlalu besar dapat menyebabkan divergensi, sedangkan nilai yang terlalu kecil dapat membuat pelatihan menjadi lambat.

5. **Ukuran Batch**:

- Ukuran batch bervariasi dari 16 hingga 512.
- Parameter ini memengaruhi kecepatan dan stabilitas pelatihan, di mana ukuran batch kecil dapat mempercepat konvergensi tetapi berpotensi meningkatkan variabilitas gradien.

Implementasi Model

Model MLP dirancang secara modular, dengan fleksibilitas untuk mengatur jumlah layer, jumlah neuron per layer, dan fungsi aktivasi. Setiap hidden layer diikuti oleh fungsi aktivasi yang dipilih, dan layer terakhir menggunakan layer linear untuk menghasilkan output berupa dua kelas (kelas 0 dan kelas 1). Optimasi dilakukan menggunakan algoritma Adam dengan *loss function* CrossEntropy yang sesuai untuk tugas klasifikasi biner.

Proses Pelatihan dan Evaluasi

Skrip ini menggunakan pendekatan sistematis dalam melatih dan mengevaluasi model:

1. **Pelatihan**:

- Data pelatihan dibagi ke dalam batch menggunakan *DataLoader* untuk mempermudah proses iterasi.
- Pada setiap epoch, model diperbarui dengan menghitung gradien dari *loss function* pada batch data, dan parameter model diperbarui menggunakan optimasi Adam.
- Optimasi Adam dipilih karena sifatnya yang adaptif, sehingga mampu menangani berbagai nilai learning rate yang digunakan dalam eksperimen.

2. **Evaluasi**:

- Setelah pelatihan selesai, model diuji pada data uji untuk menghitung akurasi prediksi.
- Prediksi diambil berdasarkan argumen dengan probabilitas tertinggi (dari output fungsi softmax di layer akhir).
- Hasil akurasi dicatat untuk setiap kombinasi parameter.

Hasil dan Analisis

Setelah semua kombinasi parameter diuji, hasil akurasi disimpan dalam daftar, diurutkan dari yang tertinggi hingga terendah. Sepuluh konfigurasi terbaik ditampilkan untuk memberikan wawasan tentang kombinasi parameter yang optimal.

Eksperimen ini memungkinkan kita untuk mengidentifikasi pola tertentu:

- **Arsitektur Hidden Layer**: Model dengan lebih banyak layer dan neuron cenderung memberikan hasil lebih baik untuk dataset kompleks. Namun, untuk dataset sederhana seperti ini, konfigurasi dengan lebih sedikit layer dan neuron dapat mencapai performa optimal dengan lebih efisien.
- **Fungsi Aktivasi**: Fungsi seperti *ReLU* biasanya unggul dalam tugas-tugas pembelajaran mendalam karena menghindari masalah gradien yang

menghilang, sementara *sigmoid* atau *tanh* dapat menghasilkan kinerja suboptimal pada data yang tidak terlalu kompleks.

- **Jumlah Epoch**: Peningkatan jumlah epoch meningkatkan akurasi hingga titik tertentu, tetapi terlalu banyak epoch dapat menyebabkan *overfitting*.
- **Learning Rate**: Nilai learning rate yang terlalu besar cenderung menyebabkan divergensi, sementara nilai yang terlalu kecil memerlukan lebih banyak epoch untuk mencapai konvergensi.
- **Ukuran Batch**: Ukuran batch kecil memungkinkan pelatihan lebih cepat tetapi kurang stabil, sedangkan ukuran batch besar memberikan pembaruan gradien yang lebih stabil.